Python

План

- Установка и работа с кодом
- Стандартные типы данных и работа с ними
  - Строки
  - Списки и кортежи
  - Множества и словари
- Функции

Работа с кодом

Запуск интерпретатора через консоль

```
python[3][.v]
```

Запуск исполняемого скрипта через консоль

```
python3  my_first_script.py [–arg_1]
```

IDE (Integrated Development Environment)

- PyCharm (https://www.jetbrains.com/ru-ru/pycharm/)

CE (Code Editor)

- VSCode (https://code.visualstudio.com)
- Atom (https://atom.io/)
- Sublime (https://www.sublimetext.com/)

Тетрадки с кодом

- Jupyter Notebook (https://jupyter.org/)
- Google Cloud Colaboratory (https://colab.research.google.com/)

In [1]:
```
# https://github.com/psf/black -- code formatter
%load_ext nb_black
```

Стандартные типы данных

https://docs.python.org/3/library/stdtypes.html
(https://docs.python.org/3/library/stdtypes.html)

In [2]:
```python
# Целое число integer -> int
# a = int(7)
a = 7

# Число с плавающей запятой -- float -> float
# b = float(3.14)
b = 3.14

# Текстовая последовательность -- string -> str
# c = str("Текст 1")
c = "Текст 1"
d = "Текст 2"
e = """Текст 3 может быть длинным и
можно переносить его на новые строки
можно переносить его на новые строки
можно переносить его на новые строки
можно переносить его на новые строки"""

# Логичекая переменная -- boolean -> bool
# f = bool(1)
f = True


# Списки list -> list
# g = list("123456789")
g = [1, 2, 3, 4, 5, 6, 7, 8, 9]

# Множества и словари -> set, dict
# h = set(["A", "B", "C"])
h = {"A", "B", "C"}

# i = dict((("key1", 1), ("key2", 2), ("key3", 3)))
i = {"key1": 1, "key2": 2, "key3": 3}

# И многие другие...
```

In [3]:
```python
# Аннотации типов
a: int = 1
b: float = 3.14
c: str = "qwerty"
```

In [4]:
```python
# строки форматирования
# https://docs.python.org/3/library/string.html#formatstrings

f"{0.2 + 0.1} > {0.3}, {0.7+0.1} < {0.7+0.1:.2f}"
```

Out[4]:
```
'0.30000000000000004 > 0.3, 0.7999999999999999 < 0.80'
```

Строки -> str

https://docs.python.org/3/library/string.html (https://docs.python.org/3/library/string.html)

In [5]:
```
print("qwerty\nasdfg!")  # unicode
print(r"qwerty\nasdfg!")  # "сырая" -- row
print(b"\xd1\x81\xd1\x82\xd1\x80\xd0\xbe\xd0\xba\xd0\xb0")  # б
print(f"qwerty -> {1+2+3} ? {b}")  # строка форматирования
```

```
qwerty
asdfg!
qwerty\nasdfg!
b'\xd1\x81\xd1\x82\xd1\x80\xd0\xbe\xd0\xba\xd0\xb0'
qwerty -> 6 ? 3.14
```

In [6]:
```
print("строка" + " и еще что-то")  # конкатенация
print("_" * 10)  # "мультиплицирование"
```

```
строка и еще что-то
__________
```

In [7]:
```
# 12345 + "строка"  # -> TypeError: мы должны работать с одним
type(str(12345) + "строка")
```

Out[7]:
```
str
```

In [8]:
```
string = "строка"
print(string[1])  # индексация с 0
print(string[-1])  # мы можем обращаться в обратном порядке
```

```
т
а
```

In [9]:
```
# string[1] = "ю"  # -> так мы делать не можем, строки неизменя
# string[100]  # -> IndexError -- не существует такого индекса
```

In [10]:
```
# Срезы
print(string[:])  # [начало:конец:шаг]
print(string[:5], string[2:10], sep=" | ")
print(string[::2], string[5:-2:3], sep=" | ")
print(string[-1::-1], string[::-1], string[1:5:-1], sep=" | ")
string[::-1]
```

```
строка
строк | рока
срк |
акортс | акортс |
```

Out[10]:
```
'акортс'
```

методы

https://docs.python.org/3/library/stdtypes.html#string-methods
(https://docs.python.org/3/library/stdtypes.html#string-methods)

In [11]:
```
string.upper(), string.lower()
```

Out[11]:
```
('СТРОКА', 'строка')
```

In [12]:
```
"Привет".isalpha(), "123".isdigit()
```

Out[12]:
```
(True, True)
```

In [13]:
```
string.replace("о", "!"), "молоко".replace("о", "*", 2)
```

Out[13]:
```
('стр!ка', 'м*л*ко')
```

In [14]:
```
"Некоторая строка".encode(), b"\xd1\x81\xd1\x82\xd1\x80\xd0\xbe
```

Out[14]:
```
(b'\xd0\x9d\xd0\xb5\xd0\xba\xd0\xbe\xd1\x82\xd0\xbe\xd1\x80\xd0\xb
0\xd1\x8f \xd1\x81\xd1\x82\xd1\x80\xd0\xbe\xd0\xba\xd0\xb0',
 'строка')
```

In [15]:
```
import math  # модуль https://docs.python.org/3/library/math.ht

print(math.e)
print(math.floor(1.2), math.ceil(1.5))
print(round(0.5), round(1.5), round(2.5), round(3.5))
```

```
2.718281828459045
1 2
0 2 2 4
```

In [16]:
```
print("1: {} 2: {:.3f} 3: {:03}".format(1, math.e, 999))
print("{1} {2} {0}".format(1, math.e, 3))
print("{first} {second} {third}".format(first=1, second=math.e,
print()

print("1)%d 2)%.2f 3)%s" % (1, 3.1415, 3))  # python 2x
print()

# https://docs.python.org/3/reference/lexical_analysis.html#for
print(f"{string} + {10+23}")
print(f"{2+3} {math.pi:.5f}")
```

```
1: 1 2: 2.718 3: 999
2.718281828459045 3 1
1 2.718281828459045 3

1)1 2)3.14 3)3

строка + 33
5 3.14159
```

Списки

https://docs.python.org/3/library/stdtypes.html#lists
(https://docs.python.org/3/library/stdtypes.html#lists)

In [17]:
```
?list

# https://docs.python.org/3/glossary.html
"""
Built-in mutable sequence.

If no argument is given, the constructor creates a new empty li
The argument must be an iterable if specified.
"""
# https://docs.python.org/3/glossary.html#term-iterable
pass
```

In [18]:
```
lst_1 = list()  # через конструктор
lst_2 = []  # пустой список
lst_3 = [1, "2", True, 4, 5]  # через запятую
lst_4 = [i for i in "123456"]  # через абстракцию списков / спи
```

In [19]:
```
import string

print(string.ascii_letters)

lst = list(string.ascii_lowercase)
print(lst[:5])
print(lst[0])

lst[-1] = -2
print(lst[:5])
lst[::-1]
```

```
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
['a', 'b', 'c', 'd', 'e']
a
['a', 'b', 'c', 'd', 'e']
```

Out[19]:
```
[-2,
 'y',
 'x',
 'w',
 'v',
 'u',
 't',
 's',
 'r',
 'q',
 'p',
 'o',
 'n',
 'm',
 'l',
 'k',
 'j',
 'i',
 'h',
 'g',
 'f',
 'e',
 'd',
 'c',
 'b',
 'a']
```

In [20]:
```
print(lst[len(lst) : len(lst) + 100])  # так можно, но будет пу
# lst[len(lst)] = "0"  # индексация с 0 -> IndexError: list ass
```

```
[]
```

методы

In [21]:
```
# Добавление в конец
lst = list("qwerty")
lst.append("1")
lst.append(2)
lst
```

Out[21]:
```
['q', 'w', 'e', 'r', 't', 'y', '1', 2]
```

In [22]:
```
# Добавление по индексу
lst.insert(2, "A")
lst.insert(100, "B")
lst
```

Out[22]:
```
['q', 'w', 'A', 'e', 'r', 't', 'y', '1', 2, 'B']
```

In [23]:
```
# Удаление по значению
lst.remove("B")
# lst.remove("C")  # ValueError: list.remove(x): x not in list
lst
```

Out[23]:
```
['q', 'w', 'A', 'e', 'r', 't', 'y', '1', 2]
```

In [24]:
```
# Удаление по индексу
removed_last = lst.pop()
removed_i = lst.pop(5)
print(lst, removed_last, removed_i)
```

```
['q', 'w', 'A', 'e', 'r', 'y', '1'] 2 t
```

In [25]:
```
# Удаление через инструкцию del
del lst[-1], lst[:2]
lst
```

Out[25]:
```
['A', 'e', 'r', 'y']
```

In [26]:
```
del lst
lst
```

```
---------------------------------------------------------------------
---------
NameError                                 Traceback (most recent c
all last)
<ipython-input-26-19d04f711f3e> in <module>
      1 del lst
----> 2 lst

NameError: name 'lst' is not defined
```

In [27]:
```python
# Очищение
lst = list("qwerty")
print(lst)
lst.clear()
print(lst)
```

```
['q', 'w', 'e', 'r', 't', 'y']
[]
```

In [28]:
```python
lst = list("qwertyq")

# Поиск
print(lst.index("y"))
# print(lst.index(1))

print(lst.count("q"))
print(lst.count(1))
# Сортировка
lst.sort()
print(lst)
```

```
5
2
0
['e', 'q', 'q', 'r', 't', 'w', 'y']
```

In [29]:
```python
lst_1 = [1, 2, 3]
lst_2 = [3, 4, 5]
lst_1.append(lst_2)
print(lst_1)

lst_1 = [1, 2, 3]
lst_2 = [3, 4, 5]
lst_1.extend(lst_2)
lst_1 += lst_2
print(lst_1)
```

```
[1, 2, 3, [3, 4, 5]]
[1, 2, 3, 3, 4, 5, 3, 4, 5]
```

In [30]:
```python
lst = list("qwertyq")
print(sorted(lst))  # -> возвращает список
```

```
['e', 'q', 'q', 'r', 't', 'w', 'y']
```

In [31]:
```python
lst[:5] + lst[:5], lst[:5] * 3, [0] * 10
```

Out[31]:
```
(['q', 'w', 'e', 'r', 't', 'q', 'w', 'e', 'r', 't'],
 ['q', 'w', 'e', 'r', 't', 'q', 'w', 'e', 'r', 't', 'q', 'w', 'e',
'r', 't'],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [32]:
```
lst_1 = [1, 2, 3]
lst_2 = [3, 4, 5]
lst_3 = [6, 7, 8]

print([lst_1, lst_2, lst_3])  # матрица
print(sum([lst_1, lst_2, lst_3], []))  # положили в один список

print(sum(lst_1), sum(lst_1, 1000))  # start = 0, start = 1000
```

```
[[1, 2, 3], [3, 4, 5], [6, 7, 8]]
[1, 2, 3, 3, 4, 5, 6, 7, 8]
6 1006
```

In [33]:
```
print(sum([1, 2], start=-3))
sum([[1, 2] * 3], [])
```

```
0
```

Out[33]:
```
[1, 2, 1, 2, 1, 2]
```

Строки <-> списки...

In [34]:
```
string = "1 2 3 4 5 6"
lst = ["1", "2", "3", "4", "5", "6"]

print(string.split())  # метод строк
print(
    "!_@".join(["1", "2", "3"])
)  # метод строк !но внутри join элементы iterable должны быть
```

```
['1', '2', '3', '4', '5', '6']
1!_@2!_@3
```

Кортежи

https://docs.python.org/3/library/stdtypes.html#tuples
(https://docs.python.org/3/library/stdtypes.html#tuples)

In [35]:
```
tpl_1 = tuple("qwerty")
tpl_2 = ()
tpl_3 = (2,)
tpl_4 = (1, 2, 3, 4)
```

In [36]:
```
# Распаковка переменных
a, b, c, d = (1, 2, 3, 4)  # для списков тоже работает
a, b, c, d = d, c, b, a
a, _, _, _ = 1, 2, 3, 4
a + b + c + d
```

Out[36]:
```
7
```

In [37]:
```
print(*tpl_4, sep="->")
print(1, 2, 3, 4)
print(tpl_4, sep="->")

# print(*[*[1] * 3] * 3, sep="\n")
```

```
1->2->3->4
1 2 3 4
(1, 2, 3, 4)
```

In [38]:
```
# Доступ
tpl_4[0], tpl_4[:]
```

Out[38]:
```
(1, (1, 2, 3, 4))
```

In [39]:
```
# Изменение и удаление элементов -> TypeError
tpl_4[0] = 8
del tpl_4[0]
```

```
---------------------------------------------------------------------
---------
TypeError                                 Traceback (most recent c
all last)
<ipython-input-39-e966a8360e47> in <module>
      1 # Изменение и удаление элементов -> TypeError
----> 2 tpl_4[0] = 8
      3 del tpl_4[0]

TypeError: 'tuple' object does not support item assignment
```

In [40]:
```
del tpl_4
```

Операторы in / not in

In [41]:
```
"abc" in "abc def ghi", "q" in list("qwerty"), 7 not in tuple((
```

Out[41]:
```
(True, True, True)
```

In [42]:
```
if "a" in "abc":
    print("In")
```

```
In
```

Циклы while,for

In [43]:
```
inc = 0
inc = inc + 1
# inc++ # c-style
inc *= 10  # *= -= ...
inc
```

Out[43]:
```
10
```

In [44]:
```
inc = 1
while inc < 10:
    print("Итерация #", inc)
    inc += 1  # не забываем увеличить
```

```
Итерация # 1
Итерация # 2
Итерация # 3
Итерация # 4
Итерация # 5
Итерация # 6
Итерация # 7
Итерация # 8
Итерация # 9
```

In [45]:
```
inc = 1
while inc < 10:
    if inc > 3:
        print("--- Условие раннего выхода из цикла ---")
        break

    print("Итерация #", inc)

    inc += 1  # не забываем увеличить
```

```
Итерация # 1
Итерация # 2
Итерация # 3
--- Условие раннего выхода из цикла ---
```

In [46]:

```
inc = 1
while inc < 10:
    print("Итерация до #", inc)
    if inc % 2:
        print("--- Тут нечетное число ---")
        inc += 1  # не забываем увеличить
        continue

    print("Итерация #", inc)

    inc += 1  # не забываем увеличить
```

```
Итерация до # 1
--- Тут нечетное число ---
Итерация до # 2
Итерация # 2
Итерация до # 3
--- Тут нечетное число ---
Итерация до # 4
Итерация # 4
Итерация до # 5
--- Тут нечетное число ---
Итерация до # 6
Итерация # 6
Итерация до # 7
--- Тут нечетное число ---
Итерация до # 8
Итерация # 8
Итерация до # 9
--- Тут нечетное число ---
```

In [47]:

```
for i in "abcd":
    print(i, end=".")
print()

for letter in [1, 2, "y"]:
    print(letter, end=",")
print()

for it in enumerate(list("qwerty")):
    print(it, end="|")

print()
for i, letter in enumerate(("qwerty")):
    print(f"{i}){letter}", end=" ")
```

```
a.b.c.d.
1,2,y,
(0, 'q')|(1, 'w')|(2, 'e')|(3, 'r')|(4, 't')|(5, 'y')|
0)q 1)w 2)e 3)r 4)t 5)y
```

In [48]:

```
?range
```

In [49]:
```
for i in range(10):
    print(i, end=" ")
print()

for i in range(5, 10):
    print(i, end=" ")
print()

for i in range(5, 10, 2):
    print(i, end=" ")
```

```
0 1 2 3 4 5 6 7 8 9
5 6 7 8 9
5 7 9
```

In [50]:
```
lst = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
len(lst)
for i in range(len(lst)):
    lst[i] = 0
    print(f"{i}-{lst[i]}", end=" ")

```

```
0-0 1-0 2-0 3-0 4-0 5-0 6-0 7-0 8-0 9-0
```

In [51]:
```
for x in range(20):
    break

for x in range(10):
    continue


print(x)
```

```
9
```

Множества

https://docs.python.org/3/library/stdtypes.html#set-types-set-frozenset
(https://docs.python.org/3/library/stdtypes.html#set-types-set-frozenset)

In [52]:
```
# на вход приходит что-то неизменяемое ->
# внутри происходит преобразование объекта по определенному пра
# на выходе, в нашем случае, мы получаем число
hash("One"), hash("1"), hash("1"), hash(1), hash(1.1)
```

Out[52]:
```
(-2641178646822081237,
 -2740927056621133663,
 -2740927056621133663,
 1,
 230584300921369601)
```

In [53]:
```
st_1 = {"1", "2", "3", "4", "1", "4", "20"}
st_2 = set("12345678")
st_1, st_2
```

Out[53]:
```
({'1', '2', '20', '3', '4'}, {'1', '2', '3', '4', '5', '6', '7', '
8'})
```

Операции и методы

In [54]:
```
st = set()
# Добавление
st.add(1)
print(st)

# Объеденить с другим и дабавить в себя
st.update({1, 2, 3, 4})
print(st)

# Удаление
st.discard(10)  # нет ошибки, если нет элемента
st.remove(1)  # -> KeyError
print(st.pop())
print(st)

# Очищение
st.clear()
print(st)

# Удаление
del st
```

```
{1}
{1, 2, 3, 4}
2
{3, 4}
set()
```

In [55]:
```
print(st_1 & st_2, st_1.intersection(st_2))  # пересечение
print(st_1 | st_2, st_1.union(st_2))  # объединение
print(st_2 - st_1, st_2.difference(st_1))  # разница
print(
    st_1 ^ st_2, st_2.symmetric_difference(st_1)
)  # не находятся сразу в двух множествах
```

```
{'2', '1', '3', '4'} {'2', '1', '3', '4'}
{'1', '3', '4', '2', '20', '6', '5', '8', '7'} {'1', '3', '4', '2'
, '20', '6', '5', '8', '7'}
{'5', '7', '8', '6'} {'5', '7', '8', '6'}
{'20', '6', '5', '8', '7'} {'5', '8', '20', '7', '6'}
```

Словари

https://docs.python.org/3/library/stdtypes.html#mapping-types-dict
(https://docs.python.org/3/library/stdtypes.html#mapping-types-dict)

In [56]:
```
# {1, 2, 3, 4, 5} -> {1:1, 2:2, 3:3, 4:4, 5:5} -- ключ: значени
dct_1 = dict([("key1", 1), ("key2", 2)])
dct_2 = {1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 2: 5}
```

In [57]:
```
# ключи должны быть хэшируемы, но значения -- неважно
{"key1": [1, 2, 3, 4], "key2": dict(), 3: (1, 2, 3), (1, 2, 3):
```

Out[57]:
```
{'key1': [1, 2, 3, 4],
 'key2': {},
 3: (1, 2, 3),
 (1, 2, 3): {'e', 'q', 'r', 't', 'w', 'y'}}
```

Операции и методы

In [58]:
```
dct = dict()
# Добавление
dct["key_1"] = 2
print(dct)

# Доступ
print(dct["key_1"])
# print(dct["key_2"])  # -> KeyError

print("->", dct.get("key_2"), dct.get("key_2", 3))  # get(key,
dct["key_2"] += dct.get("key_2", 0) + 1
print(dct)
```

```
{'key_1': 2}
2
-> None 3

---------------------------------------------------------------------
---------
KeyError                                  Traceback (most recent c
all last)
<ipython-input-58-38b0b9e7b4f7> in <module>
      9
     10 print("->", dct.get("key_2"), dct.get("key_2", 3))  # get(
key, default = None)
---> 11 dct["key_2"] += dct.get("key_2", 0) + 1
     12 print(dct)

KeyError: 'key_2'
```

In [59]:
```
"A b a A"
```

Out[59]:
```
'A b a A'
```

In [60]:
```
print(dct.keys(), dct.values(), dct.items(), sep="\n")
```

```
dict_keys(['key_1'])
dict_values([2])
dict_items([('key_1', 2)])
```

In [61]:
```
for i in dct:
    print(i, end="\t")
print()

for i in dct.keys():
    print(i, end="\t")
print()

for i in dct.values():
    print(i, end="\t")
print()

for i in dct.items():
    print(i, end="\t")
print()

for key, value in dct.items():
    print(f"{key}:{value}", end="\t")
print()
```

```
key_1
key_1
2
('key_1', 2)
key_1:2
```

List, Set, Dict Comprehensions

{открывающий символ типа данных} {что складывается} for {прозвище переменной} in {iterable} {условия} {закрывающий символ типа данных}

In [62]:
```
print([i for i in "qwerty"])
print({i for i in "qwertyqwerty"})
print({v: i for i, v in enumerate("qwertyqwerty")})

# print((i for i in range(10)))  # это немного другое...
```

```
['q', 'w', 'e', 'r', 't', 'y']
{'y', 'w', 'e', 'q', 'r', 't'}
{'q': 6, 'w': 7, 'e': 8, 'r': 9, 't': 10, 'y': 11}
```

In [63]:
```
print([i for i in range(10)])
print([i for i in range(20) if i % 2 == 0])
print([i * 2 if i % 2 == 0 else -1 for i in range(20)])
print([(i ** 2, v * i) for i, v in enumerate("qwerty")])
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
[0, -1, 4, -1, 8, -1, 12, -1, 16, -1, 20, -1, 24, -1, 28, -1, 32,
-1, 36, -1]
[(0, ''), (1, 'w'), (4, 'ee'), (9, 'rrr'), (16, 'tttt'), (25, 'yyy
yy')]
```

Тернарный оператор

In [64]:
```
# {что вернуть если true} if {условие} else {что вернуть если f
1 if 6 % 2 == 1 else -1
```

Out[64]:
```
-1
```

```
value = 0
if 6%2 == 1
    value = 1
else:
    value = -1
```

Еще пара важных фунций

In [65]:
```
?map
```

In [66]:
```
m = map(int, "123456")
next(m), next(m), next(m), next(m), next(m), next(m)  # , next(
```

Out[66]:
```
(1, 2, 3, 4, 5, 6)
```

In [67]:
```
list(map(int, "123456"))
```

Out[67]:
```
[1, 2, 3, 4, 5, 6]
```

In [68]:
```
lst = ["aaa", "bbb", "cc"]
print(lst, sep="\t")
```

```
['aaa', 'bbb', 'cc']
```

In [69]:
```
print(*map(str.upper, "qwerty"))
```

```
Q W E R T Y
```

In [70]:
```
print(*map(lambda x: x ** 2, range(10)))
```

```
0 1 4 9 16 25 36 49 64 81
```

In [71]:
```
?zip
```

In [72]:
```
print(list(zip(range(10), range(10, 0, -1), range(10))))
```

```
[(0, 10, 0), (1, 9, 1), (2, 8, 2), (3, 7, 3), (4, 6, 4), (5, 5, 5)
, (6, 4, 6), (7, 3, 7), (8, 2, 8), (9, 1, 9)]
```

In [73]:
```
# Получим пары и положим их в словарь
pairs = list(zip(range(10), range(10, 0, -1)))
print(pairs)
dict(pairs)
```

```
[(0, 10), (1, 9), (2, 8), (3, 7), (4, 6), (5, 5), (6, 4), (7, 3),
(8, 2), (9, 1)]
```

Out[73]:
```
{0: 10, 1: 9, 2: 8, 3: 7, 4: 6, 5: 5, 6: 4, 7: 3, 8: 2, 9: 1}
```

Функции

In [74]:
```
def function():  # название + аргументы, если нужны
    """Описание функции в docstring"""
    # мы что-то хотим сделать, ваш код здесь
    # мы можем что-то вернуть а можем ничего не возвращать
    pass  # оператор-заглушка == отсутствие операции


function()  # вызов функции
?function
```

In [75]:
```
def printHello():  # вывод на экран Hello
    print("Hello")


printHello()
```

```
Hello
```

In [76]:
```
def summarize(a: int, b: int) -> int:
    return a + b // a


summarize(20, 21)
```

Out[76]:
```
21
```

In [77]:
```
def function(a, b=0):  # b -- необязательный аргумент, по умолч
    return a + b


print(function(1))
print(function(2, 3))
```

```
1
5
```

In [78]:
```
def function(number):
    if number > 0:
        return "positive"
        print("---------")
    elif number < 0:
        return "negative"
        print("---------")

    return "neither positive nor negative"  # return будет в лю
    print("---------")


function(100), function(-100), function(0)
```

Out[78]:
```
('positive', 'negative', 'neither positive nor negative')
```

In [79]:
```
def function(*args):  # упаковка
    print(type(args))
    print(args)
    for i in args:
        print(i)


function(1, 2, 3, 4, 5, 6, 6, 77, 82)
# function(*[1, 2, 3])  # распаковка
```

```
<class 'tuple'>
(1, 2, 3, 4, 5, 6, 6, 77, 82)
1
2
3
4
5
6
6
77
82
```

In [80]:
```
print(*[1, 2, 3], sep="->")
print(1, 2, 3, sep="->")
```

```
1->2->3
1->2->3
```

In [81]:
```python
def function(**kwargs):  # упаковка
    print(type(kwargs))
    print(kwargs)


function(key_1=1, key_2=2)
# function(**{"key_1": 2, "key_2": 2})  # распаковка
```

```
<class 'dict'>
{'key_1': 1, 'key_2': 2}
```

In [82]:
```python
def func(a, b):
    print(a, b)


func(**{"a": 2, "b": 3})
```

```
2 3
```

In [83]:
```python
def function(*args, value):
    print(args, value)


# function(1, 2, 3, 4)
function(1, 2, 3, 4, value=5)
```

```
(1, 2, 3, 4) 5
```

In [84]:
```python
# Анонимные функции
# не требует return, обычно не "приравнивается" к переменной
lambda_function = lambda a, b: a + b
print(lambda_function(1, 2))
print((lambda x: x ** 2)(3))
```

```
3
9
```

In [85]:
```python
print(list(range(10)))
print(list(map(lambda x: x ** 2, range(10))))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```